

Real-Time Energy Disaggregation of a Distribution Feeder's Demand Using Online Learning

Gregory S. Ledva, *Student Member, IEEE*, Laura Balzano, *Member, IEEE*,
and Johanna L. Mathieu, *Member, IEEE*

Abstract

Though distribution system operators have been adding more sensors to their networks, they still often lack an accurate real-time picture of the behavior of distributed energy resources such as demand responsive electric loads and residential solar generation. Such information could improve system reliability, economic efficiency, and environmental impact. Rather than installing additional, costly sensing and communication infrastructure to obtain additional real-time information, it may be possible to use existing sensing capabilities and leverage knowledge about the system to reduce the need for new infrastructure.

In this paper, we disaggregate a distribution feeder's demand measurements into two components: 1) the demand of a population of air conditioners, and 2) the demand of the remaining loads connected to the feeder. We use an online learning algorithm, Dynamic Fixed Share (DFS), that uses the real-time distribution feeder measurements as well as models generated from historical building- and device-level data. We develop two implementations of the algorithm and conduct simulations using real demand data from households and commercial buildings to investigate the effectiveness of the algorithm. Case studies demonstrate that DFS can effectively perform online disaggregation and the choice and construction of models included in the algorithm affects its accuracy, which is comparable to that of a set of Kalman filters.

Index Terms

Online learning, machine learning, energy disaggregation, output feedback, real-time filtering

I. INTRODUCTION

Distributed energy resources (DERs) such as demand responsive electric loads and residential solar generation are becoming more common within electricity distribution networks [1], [2]. Sensing infrastructure, such as household smart meters, are also becoming more common [3]. However, distribution system operators still often lack an accurate real-time picture of overall DER characteristics such as i) the total power consumption of the air conditioners connected to a distribution feeder, or ii) the total power production of all solar panels installed on a distribution feeder.

The authors are with the Department of Electrical Engineering & Computer Science, University of Michigan, Ann Arbor, MI 48109 USA (e-mail: gsledv@umich.edu; girasole@umich.edu; jlmath@umich.edu). This research was funded by NSF Grant #ECCS-1508943.

Knowing overall DERs characteristics in real-time can help system operators, utilities, and third-party companies (such as energy efficiency and demand response providers) improve power system reliability, economic efficiency, and environmental impact. For example, 1) a system operator can better determine balancing reserve requirements by knowing the real-time production of intermittent distributed generation; 2) a utility can better plan demand response actions by knowing the weather forecast and the real-time portion of weather-dependent loads (e.g., air conditioners, heaters, dehumidifiers); 3) a demand response provider offering ancillary services to the system operator can better determine its bid capacity by knowing the real-time consumption of demand responsive loads; and 4) a demand response provider can use the real-time consumption of demand responsive loads as the feedback control signal within a load coordination algorithm.

Perfect real-time knowledge of DER characteristics requires a sensor at each of the large number (e.g., thousands) of spatially distributed devices and a communication infrastructure capable of reliably transmitting the data at the necessary frequency (e.g., every few seconds). Rather than installing additional, costly metering and communication infrastructure, in this paper, we show that it is possible to estimate real-time DER characteristics using existing sensing capabilities and some knowledge of the underlying system. Specifically, we show how to separate measurements of the net demand served by a distribution feeder into its components in real-time, using knowledge of the physical processes driving load/generation. We refer to this task as *feeder-level energy disaggregation*.

In this paper, we develop the feeder-level energy disaggregation problem framework and apply an online learning algorithm to separate the active power demand served by a feeder into the active power demand of a population of residential air conditioners and the active power demand of all other loads connected to the feeder. The algorithm [4] incorporates dynamical system models of arbitrary forms, blending aspects of machine learning and state estimation. Building upon our preliminary work [5], the contributions of this paper are to i) frame the feeder-level energy disaggregation problem, ii) adapt the machine learning algorithm in [4] to the feeder-level energy disaggregation problem, iii) develop a variation of the machine learning algorithm that allows it to include models with different underlying states, iv) demonstrate the performance of the online learning algorithm via a realistic data-driven case study, and v) compare the performance of the algorithm to that of a set of Kalman filters. Beyond [5], this paper develops a modified version of the algorithm, compares this modified implementation to a direct implementation of the algorithm, uses only real data (rather than models) to construct the feeder active power signal, uses real data to identify all load models, and compares algorithm performance to that of an aggressive benchmark as opposed to a simple prediction model.

Section II compares our problem and approach to related problems/work. Section III defines the problem framework. Section IV describes the data used to construct the underlying system, and Section V describes the models used within the algorithm. Section VI summarizes the online learning algorithm and our implementations for the feeder-level energy disaggregation problem. Section VII constructs case studies and summarizes their results. Finally, Section VIII presents the conclusions.

II. COMPARISON TO RELATED PROBLEMS AND WORK

The feeder-level energy disaggregation problem combines aspects of building-level energy disaggregation and load forecasting. Building-level energy disaggregation, also referred to as nonintrusive load monitoring [6], separates building-level demand measurements into estimates of the demand of individual or small groups of devices [7]. Disaggregation algorithms use data sampled at frequencies ranging from over 1 MHz to 0.3 mHz (i.e., hourly interval data) where higher-frequency data allows separation of more devices, in some work up to 100 [7]. The problem is not usually solved online because the goal is long-term energy efficiency decisions such as identification and replacement of faulty appliances and/or load research. Both unsupervised and supervised learning approaches have been proposed, with the latter often using models developed with submetering data.

Load forecasting predicts the total future demand within a given area over time horizons ranging from hours to years [8]. Whereas energy disaggregation typically deals with small load aggregations, load forecasting typically deals with large aggregations, e.g., thousands to millions of loads. For example, forecasting the load served by a distribution transformer is considered a “small” forecasting problem [8]. Very short term load forecasting, corresponding to intraday forecasts, generally uses fifteen minute to one hour interval data [8], [9]. Smart meter data enables offline development of detailed load models [10], which may be used online for operational decisions [11], e.g., for predicting the curtailable load [10]. However, load forecasting is typically done offline and used for planning.

In contrast to building-level energy disaggregation, feeder-level energy disaggregation involves disaggregating the demand of a large number of loads, e.g., thousands, into a small number of source signals, e.g., two. In contrast to load forecasting, feeder-level energy disaggregation involves estimating *portions* of the total demand and assumes real-time demand measurements, e.g., taken by SCADA systems at distribution substations, are available on timescales of seconds to minutes. This corresponds to relatively fast sampling for load forecasting and relatively slow sampling for building-level energy disaggregation. In contrast to both building-level energy disaggregation and load forecasting, feeder-level energy disaggregation is done online. However, much like load forecasting and some building-level energy disaggregation approaches, we assume detailed historical load data are available and used offline to construct predictive models.

Machine learning algorithms have been proposed to address a number of problems in power systems including security assessment, forecasting, and optimal operation [12]. A variety of machine learning techniques have been used to forecast load, renewable generation, and prices [13]–[17]. References [18]–[22] apply learning approaches to demand response. However, to the best of our knowledge, this is the first paper to pose and solve the feeder-level energy disaggregation problem, or to apply the approach in [4] to a power systems problem.

III. PROBLEM FRAMEWORK

We assume that a power system entity (e.g., a system operator, utility, or third-party company) has access to real-time measurements of the electricity demand served by a distribution feeder. The power system entity is interested in separating these measurements into two components in real-time, i.e., at each time-step. The first component is the power demand of a population of residential air conditioners served by the feeder, referred to as the “AC

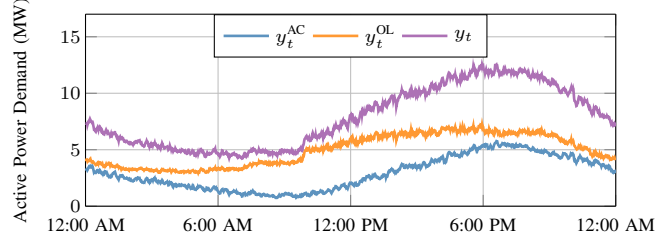


Fig. 1. Example time series of y_t and its components y_t^{OL} and y_t^{AC} .

demand.” Air conditioners generally draw power periodically to maintain a building’s indoor temperature within a range centered at a user-defined temperature set-point. The AC demand varies in time due to each air conditioner’s power cycling, weather-related influences, and building occupant influences. The second component is the power demand of the other loads on the feeder, referred to as the “OL demand,” which we assume includes both residential and commercial loads. Figure 1 displays example time series for the measured total demand y_t , the AC demand y_t^{AC} , and the OL demand y_t^{OL} over a day. We measure y_t at each time-step and try to estimate y_t^{AC} and y_t^{OL} at each time-step as each measurement arrives.

The power system entity has two distinct modes of operation. The first is the real-time estimation mode depicted in Fig. 2a. The second is the offline model generation mode, depicted in Fig. 2b. During real-time operation, we assume that the power system entity has access to active power measurements corresponding to the demand served by the distribution feeder as well as weather-related measurements for the physical area. The power measurements are time-averaged active power demands over one minute intervals, and they are the sum of the AC and OL demand. The weather-related measurements could include, for example, temperature and humidity, and can be obtained from existing weather sensors; load-specific weather monitoring is not required.

Model generation occurs offline using historical smart meter, feeder, and weather data. We assume that smart meters are installed at all houses, and they enable the collection of household- or device-level measurements at one minute intervals. The smart meters’ communication limitations [7] make real-time communication of this information infeasible, and so it is only available offline for prior days. We also assume that the power system entity has access to historical feeder and weather data. Once the models are formed, they are used along with the real-time measurements to estimate the AC and OL demand.

An online learning algorithm, Dynamic Mirror Descent (DMD) [4], uses a single model, taken from the bank of models, to generate predictions of the total demand, a loss function to penalize errors between the predicted and measured total demand, and a convex optimization formulation to adjust this prediction based on the measured total demand. Dynamic Fixed Share (DFS) [4], uses DMD within the Fixed Share Algorithm [23] to include predictions from the entire bank of models. Specifically, DFS applies DMD separately to each of the models within the bank and uses a weighting algorithm to associate a weight with each model’s adjusted prediction before combining the individual predictions into an overall estimate. In DFS, these models are weighted based on their prediction accuracy – better prediction-measurement matching leads to larger weighting and more influence in the overall prediction. The algorithm is described in detail in Section VI, but first we describe the construction of the underlying physical

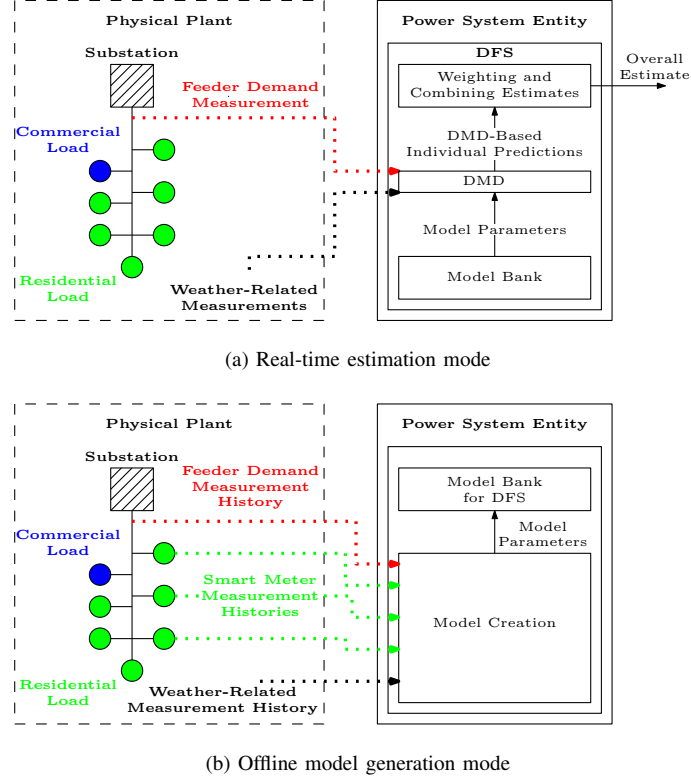


Fig. 2. Problem framework: real-time and offline modes.

system, i.e., the plant, used within the case studies and the models used within the algorithm.

IV. CONSTRUCTION OF PLANT

In this section, we detail the construction of the AC and OL demand time series and the associated weather time series over one day. These time series are used as the plant, i.e., the underlying physical system. The day consists of n^{steps} one-minute time-steps with $t = 0$ at 12:00 AM, resulting in $n^{\text{steps}} = 1440$ time-steps. Because we were unable to find sufficient data from a single location/day, we use demand and weather data from a variety of sources.

We use feeder model R5-25.00-1 from GridLAB-D's feeder taxonomy [24] to set the average residential and commercial demand on the feeder to 5.8 MW and 2.1 MW, respectively. Ignoring network losses (which, if included, would be treated as part of the OL demand), the total feeder demand measurements are the sum of the AC and OL demand, i.e., $y_t = y_t^{\text{AC}} + y_t^{\text{OL}}$, where y_t^{OL} is the sum of the other residential demand and the commercial demand $y_t^{\text{OL}} = y_t^{\text{OL, res}} + y_t^{\text{OL, com}}$. Both y_t^{AC} and $y_t^{\text{OL, res}}$ are constructed using residential data from the Pecan Street Dataport [25], while $y_t^{\text{OL, com}}$ is constructed using commercial building data provided by Pacific Gas & Electricity Company.

The residential data consists of historical 1 minute interval household- and device-level demand measurements for a set of single family homes in Austin, TX. Daily household demand signals were randomly drawn with replacement and added together until the total residential signal's mean matched that of the feeder model, resulting in 2,499 total houses. To construct the AC demand signal, we summed the demand of each household's primary air conditioner and air blower unit. Note that some houses have no/multiple air conditioner and air blower units. We assume that

only one unit per household contributes to the AC demand, resulting in 2,269 units. The remaining demand is the residential OL demand.

The commercial data consists of historical 4 second interval whole-building demand measurements from two buildings in the California Bay Area, a municipal building and a big box retail store. We summed the demand of the two buildings, and then scaled the sum by 2.61 to match the average commercial demand of the feeder model. We also down-sampled the data to 1 minute intervals by averaging the values over each minute.

The plant's weather data is constructed from two sources: 1) Pecan Street Dataport [25], and 2) National Oceanic and Atmospheric Administration (NOAA) weather station data from the National Climatic Data Center [26]. The Pecan Street weather data corresponds to the residential demand. It consists of the outdoor air temperature for Austin, TX, and it is sampled at one hour intervals. We linearly interpolate the data down to one minute intervals. The NOAA weather data corresponds to the commercial demand. It consists of outdoor temperature data from the Concord, CA weather station, sampled at one hour intervals. Again, we linearly interpolate the data down to one minute intervals. All weather data was taken from the same day as the demand data.

V. SYSTEM MODELS

In this section, we describe the models used to generate predictions of the AC and OL demands. These models are generated offline, using historical data, and then used within the online learning algorithm detailed in Section VI. The historical demand signals were constructed in the same manner as described in Section IV, using the same combination of houses as used to construct the plant signals, and excluding any atypical data such as those corresponding to holidays. The OL demand is modeled using two different linear regression methods as detailed in Section V-A, and the AC demand is modeled using several linear dynamic systems as well as a linear regression method as detailed in Section V-B.

A. OL Demand Models

We use two variations of regression models to predict the OL demand. The first model, referred to as the time-of-day (TOD) regression model, is a lookup table where an OL demand prediction is generated for each minute of the day based on the OL demand of a single day in the past. The second model is a multiple linear regression (MLR) model in which a vector of input features x_t^{OL} is used to generate an OL demand prediction for the next time-step $\hat{y}_{t+1}^{\text{OL}}$. The predictions are formed as a linear combination of the input features where the weights, or regression parameters, are generated such that they minimize the sum of the squared errors between historical measurements y_t^{OL} and their model-based predictions \hat{y}_t^{OL} . Figure 3 displays y_t^{OL} for a simulated day, several TOD regression model predictions, e.g., $\hat{y}_t^{\text{OL,Mon}}$, and the MLR regression model prediction $\hat{y}_t^{\text{OL,MLR}}$. We next describe the construction of these models.

1) *TOD Regression Models for the OL Demand:* The TOD regression model is a lookup table of the predicted OL demand for each minute of the day

$$\hat{y}_t^{\text{OL,TOD}} = \alpha_k^{\text{OL,TOD}} \quad (1)$$

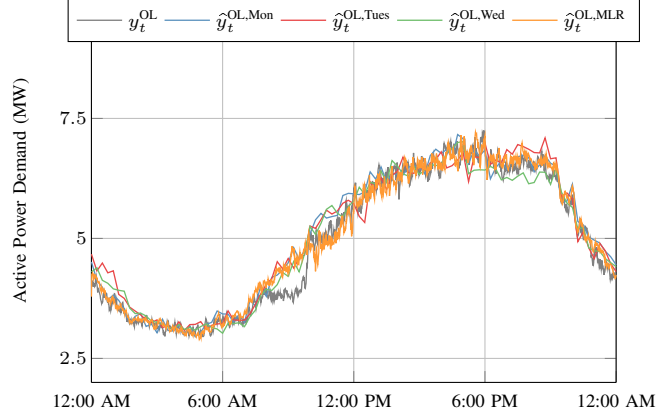


Fig. 3. Example time series of the OL demand and several OL demand model predictions.

$$= \alpha^{\text{OL},\text{TOD}} x_t^{\text{OL},\text{TOD}}. \quad (2)$$

Whereas t indexes overall time-steps, k indexes the time of day in minutes, i.e., $k = 0$ for 12:00 AM and $k = 60$ for 1:00 AM. The scalar α_k^{OL} corresponds to the predicted OL demand value for time-of-day k , $\alpha^{\text{OL},\text{TOD}}$ is a row vector containing all α_k^{OL} values, and $x_t^{\text{OL},\text{TOD}}$ is a column vector that selects the appropriate $\alpha^{\text{OL},\text{TOD}}$ based on the corresponding time of day for t . We generate $\alpha^{\text{OL},\text{TOD}}$ by smoothing the OL demand signal of a previous day using a piecewise linear and continuous, least-squares fit. Each linear segment corresponds to a fifteen minute interval of the historical data. We generate one TOD regression model for each weekday, and the models are denoted $\Phi^{\text{OL},\text{Mon}}$, $\Phi^{\text{OL},\text{Tues}}$, $\Phi^{\text{OL},\text{Wed}}$, $\Phi^{\text{OL},\text{Thurs}}$, and $\Phi^{\text{OL},\text{Fri}}$. Their corresponding predictions are $\hat{y}_t^{\text{OL},\text{Mon}}$, $\hat{y}_t^{\text{OL},\text{Tues}}$, $\hat{y}_t^{\text{OL},\text{Wed}}$, $\hat{y}_t^{\text{OL},\text{Thurs}}$, and $\hat{y}_t^{\text{OL},\text{Fri}}$, respectively.

2) *MLR Model of the OL Demand*: The MLR model of the OL demand is denoted $\Phi^{\text{OL},\text{MLR}}$, and it uses input features that include calendar-based variables, e.g., the day of the week, as well as weather-based variables, e.g., the outdoor temperature. We split the MLR model into two distinct components: one model for the commercial demand and one model for the residential OL demand since the underlying data corresponds to different geographic areas and time periods. The overall MLR model of the OL demand is then the sum of the predicted residential OL demand $\hat{y}_t^{\text{OL},\text{res}}$ and the predicted commercial demand $\hat{y}_t^{\text{OL},\text{com}}$, i.e.,

$$\hat{y}_t^{\text{OL},\text{MLR}} = \hat{y}_t^{\text{OL},\text{res}} + \hat{y}_t^{\text{OL},\text{com}} \quad (3)$$

$$= \beta^{\text{OL},\text{res}} x_t^{\text{OL},\text{res}} + \gamma^{\text{OL},\text{com}} x_t^{\text{OL},\text{com}}, \quad (4)$$

where the row vectors $\beta^{\text{OL},\text{res}}$ and $\gamma^{\text{OL},\text{com}}$ are the regression parameters for the residential OL demand and the commercial demand, respectively. The column vectors $x_t^{\text{OL},\text{res}}$ and $x_t^{\text{OL},\text{com}}$ are the corresponding input features.

The MLR model for the residential OL demand uses input features $x_t^{\text{OL},\text{res}} = \left[(x_t^{\text{TOW}})^T \quad T_t^{\text{TX}} \quad y_{t-1} \right]^T$ where x_t^{TOW} is an indicator vector for the time of week in minutes, T_t^{TX} is the outdoor temperature for Austin, TX, and y_{t-1} is the measured total demand of the previous time-step. The row vector of corresponding regression parameters is $\beta^{\text{OL},\text{res}} = \left[\beta^{\text{TOW}} \quad \beta^{\text{temp}} \quad \beta^y \right]$, where the row vector β^{TOW} consists of parameters β_n^{TOW} and n indexes the minute

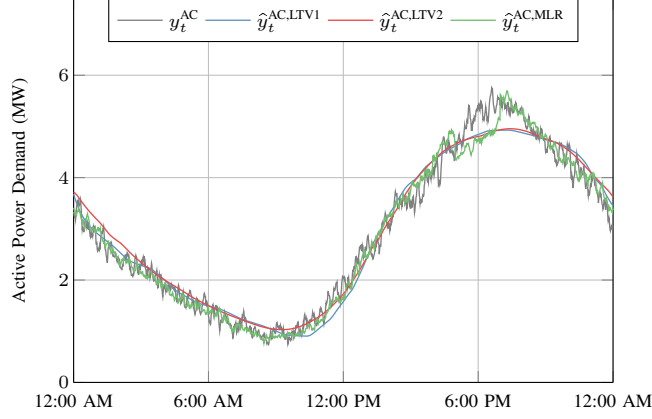


Fig. 4. Examples time series of the AC demand and several AC demand model predictions.

of the week. The indicator vector x_t^{TOW} selects the appropriate β_n^{tow} based on the corresponding time-of-week of t . The resulting form of the predictions is then

$$\hat{y}_t^{\text{OL, res}} = \beta_n^{\text{TOW}} + \beta^{\text{temp}} T_t^{\text{TX}} + \beta^y y_{t-1}. \quad (5)$$

The commercial regression model corresponds to “Baseline Method 1” from [27], which we describe here for convenience. It uses input features $x_t^{\text{OL, com}} = \left[(x_t^{\text{TOW}})^T \quad T_t^{\text{CA}} \cdot (x_t^{\text{TOW}})^T \right]^T$ where T_t^{CA} is the outdoor temperature for Concord, CA and $T_t^{\text{CA}} \cdot (x_t^{\text{TOW}})^T$ is a vector that associates the temperature to the corresponding time of week. The row vector of corresponding regression parameters is $\gamma^{\text{OL, com}} = \left[\gamma^{\text{TOW}} \quad \gamma^{\text{temp}} \right]$, where γ^{TOW} is defined similarly to β^{TOW} and row vector γ^{temp} consists of parameters γ_n^{temp} , where n indexes the time of week. The resulting form of the predictions is then

$$\hat{y}_t^{\text{OL, com}} = \gamma_n^{\text{TOW}} + \gamma_n^{\text{temp}} T_t^{\text{CA}}. \quad (6)$$

B. AC Demand Models

We use a variety of models to predict the AC demand: an MLR model, linear time invariant (LTI) system models, and linear time varying (LTV) system models. The MLR model of the AC demand is similar to the MLR model in Section V-A2 with different input features. The LTI and LTV models are versions of an aggregate model developed in [28] that predicts the portion of air conditioners that draw power versus the portion that are not drawing power. Figure 4 displays y_t^{AC} for a simulated day, several LTV model predictions, e.g., $\hat{y}_t^{\text{AC,LTV1}}$, and the MLR regression model prediction $\hat{y}_t^{\text{AC,MLR}}$. We next describe the construction of these models.

1) *MLR Model of the AC Demand*: The MLR model of the AC demand is denoted $\Phi^{\text{AC,MLR}}$, and it includes the input features $x_t^{\text{AC,MLR}} = \left[(x_t^{\text{TOW}})^T \quad T_{t-\tau^1}^{\text{TX}} \quad (T_{t-\tau^1}^{\text{TX}})^2 \quad (T_{t-\tau^1}^{\text{TX}})^3 \quad (T_{t-\tau^1}^{\text{TX}})^4 \right]^T$, where $T_{t-\tau^1}^{\text{TX}}$ is the temperature in Austin, TX from τ^1 time-steps ago and τ^1 is the time lag that maximizes the cross correlation between the historical AC demand signal and temperature signal (119 minutes for our plant). The row vector of corresponding regression parameters is $\rho^{\text{AC,MLR}} = \left[\rho^{\text{TOW}} \quad \rho^{\text{temp},1} \quad \rho^{\text{temp},2} \quad \rho^{\text{temp},3} \quad \rho^{\text{temp},4} \right]$. The resulting form of the predictions is then

$$\hat{y}_t^{\text{AC,MLR}} = \rho^{\text{AC,MLR}} x_t^{\text{AC,MLR}} \quad (7)$$

$$\begin{aligned}
&= \rho_n^{\text{TOW}} + \rho^{\text{temp},1} T_{t-\tau^1}^{\text{TX}} + \rho^{\text{temp},2} (T_{t-\tau^1}^{\text{TX}})^2 \\
&\quad + \rho^{\text{temp},3} (T_{t-\tau^1}^{\text{TX}})^3 + \rho^{\text{temp},4} (T_{t-\tau^1}^{\text{TX}})^4.
\end{aligned} \tag{8}$$

2) *LTI Model Set*: We construct a set of LTI models \mathcal{M}^{LTI} , originally developed in [28], [29]. As in [30], each model within the set captures the aggregate behavior of the population of air conditioners for a different outdoor temperature T^m and has the form

$$\hat{x}_{t+1}^{\text{LTI},m} = A^{\text{LTI},m} \hat{x}_t^{\text{LTI},m} \tag{9}$$

$$\hat{y}_t^{\text{AC,LTI},m} = C^{\text{LTI},m} \hat{x}_t^{\text{LTI},m}, \tag{10}$$

with $m \in \mathcal{M}^{\text{LTI}} = \{1, \dots, N^{\text{LTI}}\}$. The state vector $\hat{x}_t^{\text{LTI},m} \in \mathbb{R}^{N^x \times 1}$ consists of the portion of the air conditioners within each of N^x discrete states. In this paper, we use one state to represent the portion of air conditioners that are drawing power and another to represent those that are not, i.e., $N^x = 2$. The state transition matrix, $A^{\text{LTI},m} \in \mathbb{R}^{N^x \times N^x}$, is a transposed Markov transition matrix. Its entries capture the probabilities that air conditioners maintain their current state or transition to the other state during the time-step. The output matrix $C^{\text{LTI},m}$ estimates the AC demand $\hat{y}_t^{\text{AC,LTI},m}$ from the portion of air conditioners that are drawing power, i.e., $C^{\text{LTI},m} = N^{\text{ac}} \bar{P}^m \begin{bmatrix} 0 & 1 \end{bmatrix}$, where \bar{P}^m is a parameter approximating of the average power draw of air conditioners drawing power and N^{ac} is the number of air conditioners, which we assume is known.

To identify $A^{\text{LTI},m}$ and $C^{\text{LTI},m}$ for all m , we first define a set of N^{LTI} evenly spaced temperatures $\mathcal{T}^{\text{temps}} = \{T^{\min}, \dots, T^{\max}\}$ and denote the m -th temperature of the set as T^m . The difference between successive temperatures T^m and T^{m+1} is ΔT . Matrices $A^{\text{LTI},m}$ and $C^{\text{LTI},m}$ are constructed using power demand signals from each air conditioner corresponding to periods when $T^m - \frac{\Delta T}{2} \leq T_{t-\tau^1}^{\text{TX}} < T^m + \frac{\Delta T}{2}$. Some heuristics were used to exclude anomalous high or low power demand measurements. The \bar{P}^m parameter of $C^{\text{LTI},m}$ is set as the average power draw of air conditioners that are drawing power. The four entries of $A^{\text{LTI},m}$ were determined by checking whether an air conditioner 1) started drawing power, 2) stopped drawing power, 3) continued to draw power, or 4) continued to not draw power during each time-step. The occurrences for each case were counted for every air conditioner at every time-step and the totals were placed into their respective entries in $A^{\text{LTI},m}$, and then each column was normalized so that the sum of the column entries was 1. In our case studies, we construct an LTI model for each integer temperature in the set $\{74, \dots, 99\}$ °F. If the outdoor temperature lies outside of the range used to generate the models, we use the model corresponding to the closest temperature.

3) *LTV Model Based on the Delayed Temperature*: In this section, we describe how to incorporate the LTI models into an LTV model denoted $\Phi^{\text{AC,LTV1}}$ with the form

$$\hat{x}_{t+1}^{\text{LTV1}} = A_t^{\text{LTV1}} \hat{x}_t^{\text{LTV1}} \tag{11}$$

$$\hat{y}_t^{\text{AC,LTV1}} = C_t^{\text{LTV1}} \hat{x}_t^{\text{LTV1}}, \tag{12}$$

where A_t^{LTV1} and C_t^{LTV1} are generated by linearly interpolating the matrix entries based on $T_{t-\tau^1}^{\text{TX}}$. Specifically, if $T_{t-\tau^1}^{\text{TX}}$ is between temperatures T^m and T^{m+1} , A_t is formed as

$$A_t^{\text{LTV1}} = A^{\text{LTI},m} + (A^{\text{LTI},m+1} - A^{\text{LTI},m}) \frac{(T_{t-\tau^1}^{\text{TX}} - T^m)}{\Delta T} \tag{13}$$

The matrix C_t^{LTV1} is formed similarly. If the temperature lies outside of the range used to generate the models, we extrapolate using the difference between the nearest two models.

4) *LTV Model Based on a Moving Average Temperature:* We also construct an LTV model denoted $\Phi^{\text{AC,LTV2}}$ that uses a moving average of the historical outdoor temperature

$$\bar{T}_t^{\text{TX}} = \frac{1}{\tau^w} \sum_{k=t-\tau^w+1}^t T_k^{\text{TX}},$$

where τ^w is number of time-steps used to compute the moving average. We chose τ^w to be the value that maximizes the cross correlation between the historical moving average temperature and the historical AC demand signal (270 minutes for our plant). We construct the underlying LTI models as in Section V-B2 except that we divide the data based on the historical moving average temperature rather than the delayed temperature. We construct the LTV model as in Section V-B3 and, again, use extrapolation for temperatures outside of the range used to generate the models. We denote the resulting predictions as $\hat{y}_t^{\text{AC,LTV2}}$.

VI. ONLINE LEARNING ALGORITHM

In this section, we first summarize the DFS algorithm developed in [4] and then describe two algorithm implementations, one inspired by DFS but not a direct implementation and one a direct implementation. DFS incorporates DMD, also developed in [4], into the Fixed Share algorithm originally developed in [23]. The Fixed Share algorithm combines a set of predictions that are generated by independent experts, e.g., models, into an estimate of the system parameter using the experts' historical accuracy with respect to observations of the system. DMD extends the traditional online learning framework by incorporating dynamic models, enabling the estimation of time-varying system parameters (or states). DFS uses DMD, applied independently to each of the models, as the experts within the Fixed Share algorithm.

A. The DFS Algorithm

The objective of DFS is to form an estimate $\hat{\theta}_t \in \Theta$ of the dynamic system parameter $\theta_t \in \Theta$ at each discrete time-step t where $\Theta \subset \mathbb{R}^p$ is a bounded, closed, convex feasible set. The underlying system produces observations, i.e., measurements, $y_t \in \mathcal{Y}$ at each time-step after the prediction has been formed, where $\mathcal{Y} \subset \mathbb{R}^q$ is the domain of the measurements. From a control systems perspective, this is equivalent to a state estimation problem where θ_t is the system state.

DFS utilizes a set of N^{mdl} models defined as $\mathcal{M}^{\text{mdl}} = \{1, \dots, N^{\text{mdl}}\}$ to generate the estimate $\hat{\theta}_t$. To do this, DFS applies the DMD algorithm to each model, forming predictions $\hat{\theta}_t^m$ for each $m \in \mathcal{M}^{\text{mdl}}$. DMD is executed in two steps (similar to a discrete-time Kalman filter): 1) an observation-based update incorporates the new measurement into the parameter prediction, and 2) a model-based update advances the parameter prediction to the next time-step. DFS then uses the Fixed Share algorithm to form the estimate $\hat{\theta}_t$ as a weighted combination of the individual model's DMD-based predictions. A weighting algorithm computes the weights based on each model's historical accuracy with respect to the observations y_t , and models that perform poorly have less influence in the overall estimate.

The DFS algorithm is [4]

$$\tilde{\theta}_t^m = \arg \min_{\theta \in \Theta} \eta^s \langle \nabla \ell_t(\hat{\theta}_t^m), \theta \rangle + D(\theta \| \hat{\theta}_t^m) \quad (14)$$

$$\hat{\theta}_{t+1}^m = \Phi^m(\tilde{\theta}_t^m) \quad (15)$$

$$w_{t+1}^m = \frac{\lambda}{N^{\text{mdl}}} + (1 - \lambda) \frac{w_t^m \exp(-\eta^r \ell_t(\hat{\theta}_t^m))}{\sum_{j=1}^{N^{\text{mdl}}} w_t^j \exp(-\eta^r \ell_t(\hat{\theta}_t^j))} \quad (16)$$

for each $m \in \mathcal{M}^{\text{mdl}}$, and

$$\hat{\theta}_{t+1} = \sum_{m \in \mathcal{M}^{\text{mdl}}} w_{t+1}^m \hat{\theta}_{t+1}^m, \quad (17)$$

where each term is defined below. DMD is applied to each model in (14) and (15) to form the expert predictions, where (14) is a convex program that constructs the measurement-based update to the previous prediction and (15) is the model-based advancement of the adjusted prediction. The Fixed Share algorithm consists of (16) and (17), where (16) computes the weights and (17) computes the estimate as a weighted combination of the individual experts' estimates. We note that the Fixed Share algorithm's updates are independent of the dynamics and only use the experts' predictions and their resulting losses.

In (14), we minimize over the variable θ , $\eta^s > 0$ is a step-size parameter, and $\langle \cdot, \cdot \rangle$ is the standard dot product. The value $\nabla \ell_t(\hat{\theta}_t)$ is a subgradient of the convex loss function $\ell_t : \Theta \rightarrow \mathbb{R}$. The loss function $\ell_t(\cdot)$ penalizes the error between the predicted and observed values y_t using a known, possibly time-varying, function $h_t : \Theta \rightarrow \mathcal{Y}$ that maps θ_t to an observation, i.e., $y_t = h_t(\theta_t)$, to form predictions of the measurements. An example loss function is $\ell_t(\hat{\theta}_t) = \|C\hat{\theta}_t - y_t\|_2^2$ where the matrix C is $h_t(\cdot)$. In (15), the function $\Phi^m(\cdot)$ applies model m to advance the adjusted estimate $\tilde{\theta}_t^m$ in time, and each $\Phi^m(\cdot)$ can have arbitrary form, e.g., be a nonlinear function, and can have time-varying model parameters. In (16), the weight associated with model m at time-step t is w_t^m , $\lambda \in (0, 1)$ dictates the amount of weight that is shared amongst models, and η^r is a user-defined parameter that influences switching speed. The weight for model m is based on the loss of each model and the total loss of all models.

In (14), the term $\eta^s \langle \nabla \ell_t(\hat{\theta}_t), \theta \rangle$ captures the alignment of the variable θ with the positive gradient of $\ell_t(\hat{\theta}_t)$. To minimize this term alone, we would choose θ to be exactly aligned with the negative gradient direction. The term $D(\theta \| \hat{\theta}_t)$ is a Bregman divergence that penalizes the deviation between the new variable θ and the old variable $\hat{\theta}_t$. Note that, for simplicity, we have excluded regularization within (14), which DMD readily incorporates [4].

B. Algorithm Implementations

In this section, we describe two algorithm implementations to update the expert predictions. First, we describe an implementation that uses the concept of DMD, i.e., model-based predictions and measurement-based update to those predictions, but it is not a direct implementation of DMD. This method treats the models as black boxes and adjusts only their output, i.e., the OL and AC demand predictions, using the measured and predicted total feeder demand. Second, we describe a direct implementation of DMD. In contrast to the previous method, this method updates the state x_t of the LTI and LTV AC demand models. In the following, the total demand model

is $\Phi(\cdot) = \{\Phi^{\text{AC}}(\cdot), \Phi^{\text{OL}}(\cdot)\}$ where $\Phi^{\text{AC}}(\cdot)$ is an AC demand model and $\Phi^{\text{OL}}(\cdot)$ is an OL demand model. Their respective predictions are \hat{y}_t^{AC} and \hat{y}_t^{OL} .

1) *Update Method 1:* We develop a variation of the DMD algorithm that allows us to include a diverse set of models. The models used within this paper have different underlying parameters, dynamic variables, and/or structures, which makes it difficult to define a common θ_t across all of the models used. As a result, we desire a formulation that is similar to DMD, but that allows us to adjust the demand predictions directly, rather than applying the updates to quantities influencing the demand predictions.

To achieve this, we modify the DMD formulation to

$$\hat{\kappa}_{t+1} = \arg \min_{\theta \in \Theta} \eta^s \left\langle \nabla \ell_t(\hat{\theta}_t), \theta \right\rangle + D(\theta \| \hat{\kappa}_t) \quad (18)$$

$$\check{\theta}_{t+1} = \Phi(\check{\theta}_t) \quad (19)$$

$$\hat{\theta}_{t+1} = \check{\theta}_{t+1} + \hat{\kappa}_{t+1}. \quad (20)$$

The AC and OL demand models generate their predictions independently from one another, and so (20) can be rewritten as

$$\hat{\theta}_{t+1} = \Phi(\check{\theta}_t) + \hat{\kappa}_{t+1} \quad (21)$$

$$= \begin{bmatrix} \Phi^{\text{AC}}(\check{\theta}_t) \\ \Phi^{\text{OL}}(\check{\theta}_t) \end{bmatrix} + \hat{\kappa}_{t+1}. \quad (22)$$

The convex program (18) is now used to update a value $\hat{\kappa}_t$ that accumulates the deviation between the predicted and actual measurements. The model-based update (19) computes an open-loop prediction $\check{\theta}_{t+1}$, meaning that the measurements do not influence $\check{\theta}_{t+1}$. The measurement-based updates and model-based, open-loop predictions are combined in (20). In contrast, DMD uses a closed-loop model-based update where the convex program adjusts the parameter estimate to $\check{\theta}_t$, which is used to compute the next parameter estimate $\hat{\theta}_{t+1}$.

In this method, we define θ_t as the AC and OL demand, i.e., $\theta_t = \begin{bmatrix} y_t^{\text{AC}} & y_t^{\text{OL}} \end{bmatrix}^T$. The mapping from the parameter to the measurement is $h_t(\theta_t) = C_t \theta_t$ where the matrix $C_t = \begin{bmatrix} 1 & 1 \end{bmatrix}$. While the mapping and matrix are time-invariant, they may be time-varying in Section VI-B2, and so we use the more general notation. We choose the loss function as $\ell_t(\hat{\theta}_t) = \frac{1}{2} \|C_t \hat{\theta}_t - y_t\|_2^2$ and the divergence as $D(\theta \| \hat{\theta}_t) = \frac{1}{2} \|\theta - \hat{\theta}_t\|_2^2$. We can then write (18) in closed form as

$$\hat{\kappa}_{t+1} = \hat{\kappa}_t + \eta^s C_t^T (y_t - C_t \hat{\theta}_t). \quad (23)$$

2) *Update Method 2:* This method applies only to dynamic system models with dynamic states, i.e., in this paper the LTI or LTV AC demand models, which have dynamic states x_t . We set $\theta_t = \begin{bmatrix} x_t^T & y_t^{\text{OL}} \end{bmatrix}^T$, where x_t is $\hat{x}_t^{\text{LTI},m}$ in (9), \hat{x}_t^{LTV1} in (11), or \hat{x}_t^{LTV2} in an update equation similar to (11). The mapping from the parameter to the measurement is then $C_t = \begin{bmatrix} C_t^{\text{AC}} & 1 \end{bmatrix}$ where C_t^{AC} is the output matrix of the LTI or LTV AC demand model, i.e., $C_t^{\text{LTI},m}$, C_t^{LTV1} , or C_t^{LTV2} . Defining the system parameter in this way allows us to update the dynamic states of the LTI and LTV AC demand models, rather than just the output as in Update Method 1.

The model-based update is then

$$\hat{\theta}_{t+1} = \begin{bmatrix} \Phi^{\text{AC}}(\tilde{\theta}_t) \\ \Phi^{\text{OL}}(\check{\theta}_t) \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix} \hat{\kappa}_{t+1}, \quad (24)$$

where we update the AC demand model using the adjusted parameter estimate, as in DMD. Because the OL demand models do not include dynamic states, we continue to update their estimates according to Update Method 1. We again use (23) as the measurement-based update.

VII. CASE STUDIES

In this section, we investigate the performance of DFS on our problem. First, we define the scenarios, describe the benchmark, summarize the data and parameters settings, and define the performance metric used in the case studies. Following this, we present the results.

A. Scenario Definitions

We construct a number of scenarios using three sets of models and the two implementation methods.

We define the three sets of models for use within DFS:

- 1) $\mathcal{M}^{\text{Full}}$, all of the models developed in Section V, i.e., every combination of AC and OL demand models from the AC demand model set $\mathcal{M}^{\text{AC,Full}} = \{\mathcal{M}^{\text{LTI}}, \Phi^{\text{AC,MLR}}, \Phi^{\text{AC,LTV1}}, \Phi^{\text{AC,LTV2}}\}$ and the OL demand model set $\mathcal{M}^{\text{OL}} = \{\Phi^{\text{OL,Mon}}, \Phi^{\text{OL,Tues}}, \Phi^{\text{OL,Wed}}, \Phi^{\text{OL,Thurs}}, \Phi^{\text{OL,Fri}}, \Phi^{\text{OL,MLR}}\}$;
- 2) \mathcal{M}^{Red} , a reduced set that excludes the LTI models, which are not accurate over the course of the day, i.e., every combination of AC and OL demand models from the reduced AC demand model set $\mathcal{M}^{\text{AC,Red}} = \{\Phi^{\text{AC,MLR}}, \Phi^{\text{AC,LTV1}}, \Phi^{\text{AC,LTV2}}\}$ and \mathcal{M}^{OL} ; and
- 3) \mathcal{M}^{KF} , a further reduced set that excludes the MLR AC demand model, which can not be used in a Kalman filter, i.e., every combination of AC and OL demand models from the further reduced AC demand model set $\mathcal{M}^{\text{AC,KF}} = \{\Phi^{\text{AC,LTV1}}, \Phi^{\text{AC,LTV2}}\}$ and \mathcal{M}^{OL} .

Since the DMD implementation, Update Method 2, is only applicable to the LTI and LTV AC demand models, simulations corresponding to Update Method 2 apply the method to all applicable model combinations and otherwise use Update Method 1, e.g., if the AC demand model is the MLR model.

B. Kalman filter benchmark

A set of Kalman filters are used to establish a benchmark for the DFS algorithm. A Kalman filter uses measurements, an assumed system model, and known statistics of random variables, which are assumed to be zero-mean and normally distributed, to estimate the value of dynamic system parameters, i.e., the system state, at each time-step. Additional background on Kalman filters can be found in [31].

We use the LTV AC demand models within the Kalman filters. For each LTV model, the covariance of the process noise, which includes modeling error, is computed using a historical week of data, where the true state is generated using the measured AC demand and the LTV model's matrices. The Kalman filter estimates the state of

TABLE I
PARAMETER η^s USED IN THE DFS SCENARIOS

Model Set	$\mathcal{M}^{\text{Full}}$	$\mathcal{M}^{\text{Full}}$	\mathcal{M}^{Red}	\mathcal{M}^{Red}	\mathcal{M}^{KF}	\mathcal{M}^{KF}
Update Method	1	2	1	2	1	2
η^s	0.013	0.015	0.4	0.013	0.4	0.5

the AC demand model, i.e., $\theta_t = x_t$ where x_t is \hat{x}_t^{LTV1} or \hat{x}_t^{LTV2} , using output pseudo-measurements of the AC demand $\hat{y}_t^{\text{AC}} = y_t - \hat{y}_t^{\text{OL}}$. We assume that y_t is noise-free, but \hat{y}_t^{AC} is noisy due to OL demand prediction error. The covariance of the measurement errors depends on the OL demand model used, and is computed for each model using a week of historical errors.

We simulate one Kalman filter for each model pair in the set \mathcal{M}^{KF} . We compare the performance of the DFS algorithm to that of the best Kalman filter (BKF), which takes the lowest *ex post* RMSE value achieved by a Kalman filter within the set \mathcal{M}^{KF} , and that of the average Kalman filter (AKF), which is the average RMSE value across all of the Kalman filters.

C. Data

We test the methods on data from August 3-5, 10-14, 17, and 18, where the commercial data is from 2009 and the residential data is from 2015. Note that the dates in both years pertain to the same days of the week. We excluded weekends and days on which demand response actions were taken by the commercial buildings. The household and commercial demand data for August 3 were used to determine the set of houses included on the feeder, and this was set was used to construct all other demand signals as described in Section IV.

To generate the MLR regression models of the AC and OL demand, we use data from June 24 to August 2, 2015 and commercial data from June 24 to August 2, 2009. The LTI and LTV models of the AC demand were constructed using device-level data from individual air conditioners from May 2 to August 2, 2015. The TOD regression models and Kalman filter covariance matrices were generated using data from the week preceding August 3.

D. Parameters Settings

Table I summarizes the settings of η^s , and we set $\lambda = \eta^r = 1.0 \times 10^{-5}$ across all simulations. Parameter λ dictates the amount of weight shared amongst the models, where high values, i.e., near 1, force the DFS algorithm to generate estimates that are close to an average of the predictions of all models. By using a low value, i.e., near 0, a single model can dominate the estimate, if one model proves to be more accurate than the rest. Parameter η^r was roughly tuned using the simulation for August 3. Rather than tuning to optimize the RMSE for the day, the parameter was tuned to achieve qualitative characteristics of fast switching between models without over-fitting, i.e., allowing the weights to become erratic due to noise. Furthermore, the optimal η^r for a given day will generally not be optimal across all days, and so tuning to achieve the desired qualitative characteristics is appropriate. The η^s parameter was also tuned in this manner.

TABLE II
RMSE (kW) FOR THE DFS SCENARIOS AVERAGED OVER 10 SIMULATED DAYS

Model Set	$\mathcal{M}^{\text{Full}}$	$\mathcal{M}^{\text{Full}}$	\mathcal{M}^{Red}	\mathcal{M}^{Red}	\mathcal{M}^{KF}	\mathcal{M}^{KF}
Update Method	1	2	1	2	1	2
Total Demand	196.8	195.6	100.0	199.1	99.7	96.7
AC Demand	308.0	390.2	220.6	230.9	226.5	272.6
OL Demand	291.4	381.6	222.3	216.2	228.2	276.9

E. Performance metric

The root mean square error (RMSE) is used to quantify the performance of each scenario, where the RMSE for an arbitrary quantity ψ_t and its prediction $\hat{\psi}_t$ over n^{steps} time-steps is defined as $\epsilon^{\text{RMSE}} = \sqrt{\sum_{t=1}^{n^{\text{steps}}} (\psi - \hat{\psi}_t)^2 / n^{\text{steps}}}$.

F. Results

Figure 5 depicts time series for the August 17 simulation with \mathcal{M}^{Red} while using Update Method 1. The time series include the true total demand, OL demand, and AC demand along with the DFS estimates of each and the BKF estimate of the AC demand. Figure 6 shows the evolution of the dominant model weights. The weights of the remaining models are summed and referred to as “Other Models.” In this scenario, the total demand is accurately separated into its AC demand and OL demand components in real-time, where the RMSE of the total demand, AC demand, and OL demand is 93.2 kW, 151.0 kW, and 150.8 kW, respectively. In this scenario, DFS produces a more accurate AC demand estimate than BKF, which has an AC demand RMSE of 177.3 kW, but this is not always the case. The RMSE of the AC demand for AKF is 214.0 kW. The majority of the weight is initially given to the “Other Models,” because we initialize all models with the same weight. As the simulation progresses, the weight shifts between different model combinations. Since the combinations $\{\Phi^{\text{AC,LTV2}}, \Phi^{\text{OL,MLR}}\}$ and $\{\Phi^{\text{AC,LTV1}}, \Phi^{\text{OL,MLR}}\}$ perform best, they eventually earn more weight and dominate the predictions.

Table II presents the average RMSE across the simulated days for the total demand, the AC demand, and the OL demand for each DFS scenario. For comparison, BKF achieves an average RMSE of 195.3 kW for the AC demand and AKF achieves an average RMSE of 259.4 kW for the AC demand. The models corresponding to the BKF vary from day to day. To demonstrate the value of the measurement-based updates, we generated results for the full set of days using the model set \mathcal{M}^{Red} and with $\eta^s = 0$; the measurement-based update is irrelevant with this parameter setting. The resulting total demand, AC demand, and OL demand RMSEs were 260.4 kW, 254.2 kW, and 245.2 kW, respectively. These are significant increases over the DFS scenarios using \mathcal{M}^{Red} .

The scenarios using $\mathcal{M}^{\text{Full}}$ have significantly higher AC demand RMSEs than the simulations using \mathcal{M}^{Red} as well as the BKF and AKF simulations. Each of the LTI models may be accurate for a portion of the day when the AC demand is near the steady-state demand of the particular model. However, as the AC demand changes due to changes in the outdoor temperature, a given LTI model will become highly inaccurate. The DFS algorithm takes time to shift weight from the inaccurate model that was heavily weighted to the new model, and this results in increased RMSE. Eliminating these “bad models”, by using \mathcal{M}^{Red} rather than $\mathcal{M}^{\text{Full}}$, eliminates this issue.

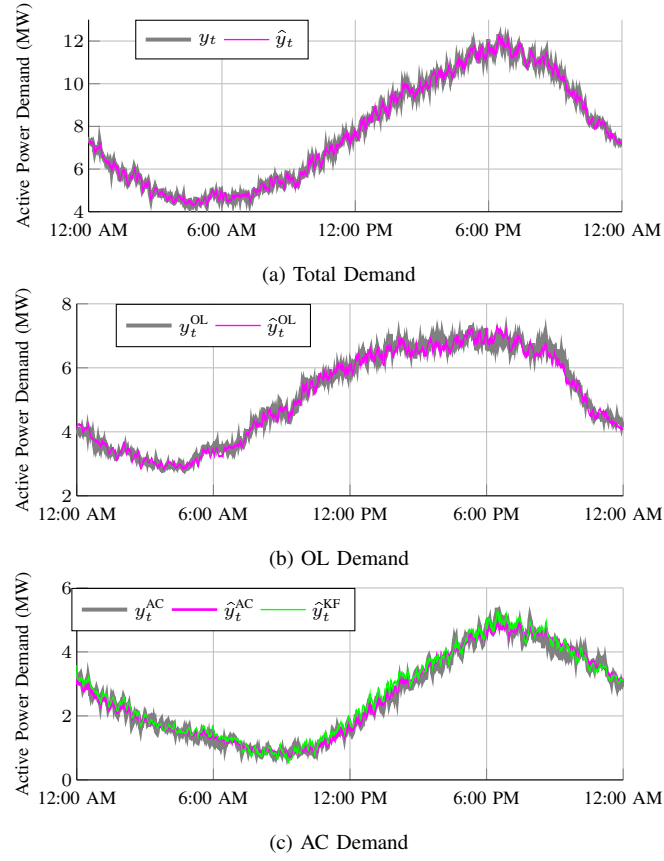


Fig. 5. Time series of the total, OL, and AC demands versus their DFS estimates from the August 17 simulation with \mathcal{M}^{Red} while using Update Method 1. The best Kalman filter (BKF) estimate of the AC demand is also shown.

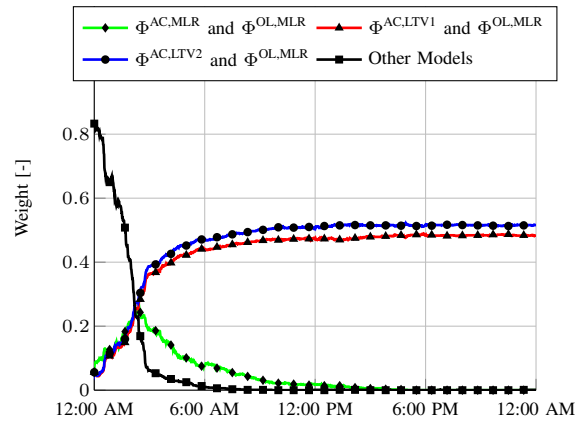


Fig. 6. Model weights time series from the August 17 simulation with \mathcal{M}^{Red} while using Update Method 1.

The scenarios using \mathcal{M}^{Red} generally do better, in terms of AC demand RMSE, than AKF and worse than BKF. On some simulated days DFS also outperforms BKF, as was shown in Fig. 5. Within this set of simulations, Update Method 2 results in higher AC demand RMSE than Update Method 1. The increased RMSE in Update Method 2 versus Update Method 1 can be explained due to the usage of only two discrete states within the LTV models. Specifically, the states reach their steady-state values rapidly, and so the measurement-based updates to the state do not persist for very long, whereas the measurement-based updates to the output used in Update Method 1 do. Using LTV models with more discrete states may allow Update Method 2 to achieve better RMSE, but this would complicate system identification.

Finally, the scenarios with \mathcal{M}^{KF} result in larger AC demand RMSE than those with \mathcal{M}^{Red} . The MLR model of the AC demand is often weighted heavily in the \mathcal{M}^{Red} simulations, especially for Update Method 2. Given this, it makes sense that excluding this model would result in increased RMSE.

VIII. CONCLUSIONS

In this paper, we applied an online learning algorithm, DFS, which uses DMD together with the Fixed Share algorithm, to estimate the real-time AC demand on a distribution feeder using feeder demand measurements, weather data, and system models. Two implementations of algorithms based on DMD were developed and compared via simulations. Our results showed that DFS can effectively estimate the real-time AC demand on a feeder. DFS achieved lower AC demand RMSE than the average across a set of Kalman filters. When selecting the most accurate Kalman filter *ex post*, DFS generally results in larger RMSE. However, DFS learns the most accurate model, or combination of models, in real-time whereas the best Kalman filter can only be chosen after the simulation. The performance of DFS depends heavily on the inclusion of models within its set. Including models that are inaccurate for majority of the day degraded the algorithm performance as did removing models that were frequently weighted heavily.

Future work will develop improved AC demand models, investigate the relationship between the DMD and Kalman filter algorithms, and incorporate active control into the problem framework. Including more than two states into the LTV models may allow these models to be more effective, but will complicate system identification. DMD and Kalman filtering have structural similarities and further establishing these similarities will help connect the online learning and state estimation fields. Finally, taking advantage of the ability to manipulate the AC demand through active control of demand response resources can enable better disaggregation.

ACKNOWLEDGMENTS

We thank the PG&E Company for the commercial building electric load data.

REFERENCES

- [1] GTM Research/SEIA: U.S. Solar Market Insight, “Solar market insight report 2015 Q1,” 2015. [Online]. Available: <http://www.seia.org/research-resources/solar-market-insight-report-2015-q1>
- [2] Navigant Research, “Direct load control and dynamic pricing programs, DR markets, and DR management systems for residential customers: Global market analysis and forecasts,” 2015. [Online]. Available: <https://www.navigantresearch.com/research/residential-demand-response#>

- [3] M. P. Lee, O. Aslam, B. Foster, S. Hou, D. Kathan, C. Pechman, and C. Young, "Assessment of Demand Response & Advanced Metering," Federal Energy Regulatory Commission, Staff Report, December 2015, <https://www.ferc.gov/legal/staff-reports/2015/demand-response.pdf>.
- [4] E. C. Hall and R. M. Willett, "Online convex optimization in dynamic environments," *IEEE Journal of Selected Topics in Signal Processing*, vol. 9, no. 4, pp. 647–662, 2015.
- [5] G. S. Ledva, L. Balzano, and J. L. Mathieu, "Inferring the behavior of distributed energy resources with online learning," in *Allerton Conference on Communication, Control, and Computing*, 2015, pp. 187–194.
- [6] M. E. Berges, E. Goldman, H. S. Matthews, and L. Soibelman, "Enhancing electricity audits in residential buildings with nonintrusive load monitoring," *Journal of Industrial Ecology*, vol. 14, no. 5, pp. 844–858, 2010.
- [7] K. C. Armel, A. Gupta, G. Shrimali, and A. Albert, "Is disaggregation the holy grail of energy efficiency? The case of electricity," *Energy Policy*, vol. 52, pp. 213–234, 2013.
- [8] T. Hong and S. Fan, "Probabilistic electric load forecasting: A tutorial review," *International Journal of Forecasting*, vol. 32, no. 3, pp. 914–938, 2016.
- [9] T. Hong, "Short term electric load forecasting," Ph.D. dissertation, North Carolina State University, 2010.
- [10] T. Byers, "How Comverge is using machine learning to improve demand response forecasts," February 2016, Comverge: Expert Insights. [Online]. Available: <http://blog.comverge.com/expert-insights/how-comverge-is-using-machine-learning-to-improve-demand-response-forecasts/>
- [11] J. W. Taylor, "An evaluation of methods for very short-term load forecasting using minute-by-minute British data," *International Journal of Forecasting*, vol. 24, no. 4, pp. 645–658, 2008.
- [12] N. Hatzigargyriou, "Machine learning applications to power systems," in *Machine Learning and Its Applications*. Springer, 2001, pp. 308–317.
- [13] M. Negnevitsky, P. Mandal, and A. K. Srivastava, "Machine learning applications for load, price and wind power prediction in power systems," in *Intelligent System Applications to Power Systems*, 2009, pp. 1–6.
- [14] D. Niu, Y. Wang, and D. D. Wu, "Power load forecasting using support vector machine and ant colony optimization," *Expert Systems with Applications*, vol. 37, no. 3, pp. 2531–2539, 2010.
- [15] M.-G. Zhang, "Short-term load forecasting based on support vector machines regression," in *2005 International Conference on Machine Learning and Cybernetics*, vol. 7. IEEE, 2005, pp. 4310–4314.
- [16] N. Sharma, P. Sharma, D. Irwin, and P. Shenoy, "Predicting solar generation from weather forecasts using machine learning," in *IEEE International Conference on Smart Grid Communications*, 2011, pp. 528–533.
- [17] T. Teo, T. Logenthiran, and W. Woo, "Forecasting of photovoltaic power using extreme learning machine," in *2015 IEEE Innovative Smart Grid Technologies-Asia*, 2015.
- [18] J. A. Taylor and J. L. Mathieu, "Index policies for demand response," *IEEE Transactions on Power Systems*, vol. 29, no. 3, pp. 1287–1295, 2014.
- [19] D. Kalathil and R. Rajagopal, "Online learning for demand response," in *Allerton Conference on Communication, Control, and Computing*, 2015, pp. 218–222.
- [20] F. Ruelens, B. Claessens, S. Vandael, B. De Schutter, R. Babuska, and R. Belmans, "Residential demand response of thermostatically controlled loads using batch reinforcement learning," *IEEE Transactions on Smart Grid*, 2016.
- [21] K. Khezeli and E. Bitar, "Risk-sensitive learning and pricing for demand response," *arXiv preprint arXiv:1611.07098*, 2016.
- [22] A. Lesage-Landry and J. A. Taylor, "Learning to shift thermostatically controlled loads," in *Proceedings of the 50th Hawaii International Conference on System Sciences*, 2017.
- [23] M. Herbster and M. K. Warmuth, "Tracking the best expert," *Machine Learning*, vol. 32, no. 2, pp. 151–178, 1998.
- [24] K. P. Schneider, Y. Chen, D. P. Chassin, R. G. Pratt, D. W. Engel, and S. E. Thompson, "Modern grid initiative distribution taxonomy final report," Pacific Northwest National Laboratory (PNNL), Richland, WA (US), Tech. Rep., 2008.
- [25] Pecan Street Inc., "Dataport," 2016.
- [26] NOAA, "NNDC Climatic Data Online," 2009, Satellite and Information Service National Climate Data Center. [Online]. Available: <http://www7.ncdc.noaa.gov/CDO/dataproduct>
- [27] J. Mathieu, A. Gadgil, D. Callaway, P. Price, and S. Kiliccote, "Characterizing the response of commercial and industrial facilities to dynamic pricing signals from the utility," in *Proceedings of ASME 2010 4th International Conference on Energy Sustainability*, Phoenix, AZ, May 2010.

- [28] J. Mathieu, S. Koch, and D. Callaway, "State estimation and control of electric loads to manage real-time energy imbalance," *IEEE Transactions on Power Systems*, vol. 28, no. 1, pp. 430–440, 2013.
- [29] K. Kalsi, M. Elizondo, J. Fuller, S. Lu, and D. Chassin, "Development and validation of aggregated models for thermostatic controlled loads with demand response," in *Proceedings of Hawaii International Conference on Systems Science*, Wailea, HI, 2012.
- [30] J. L. Mathieu, M. Kamgarpour, J. Lygeros, G. Andersson, and D. S. Callaway, "Arbitraging intraday wholesale energy market prices with aggregations of thermostatic loads," *IEEE Transactions on Power Systems*, vol. 30, no. 2, pp. 763–772, 2015.
- [31] M. S. Grewal and A. P. Andrews, *Kalman filtering*. Wiley, 2015.